PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
H04N

A2

(11) International Publication Number: WO 98/57489

(43) International Publication Date: 17 December 1998 (17.12.98)

(21) International Application Number: PCT/US98/11485

(22) International Filing Date: 8 June 1998 (08.06.98)

(30) Priority Data:
08/872,480 9 June 1997 (09.06.97) US

(71) Applicant: METALITHIC SYSTEMS, INC. [US/US]; Suite 206, 3 Harbor Drive, Sausalito, CA 94965 (US).

(72) Inventors: O'REILLY, James, M.; Suite 206, 3 Harbor Drive, Sausalito, CA 94965 (US). EIGEN, Daryl; 2280 Paradise Drive, Tiburon, CA 94920 (US).

(74) Agents: KORN, Martin et al.; Locke Purnell Rain Harrell; Suite 2200, 2200 Ross Avenue, Dallas, TX 75201 (US).

(81) Designated States: JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

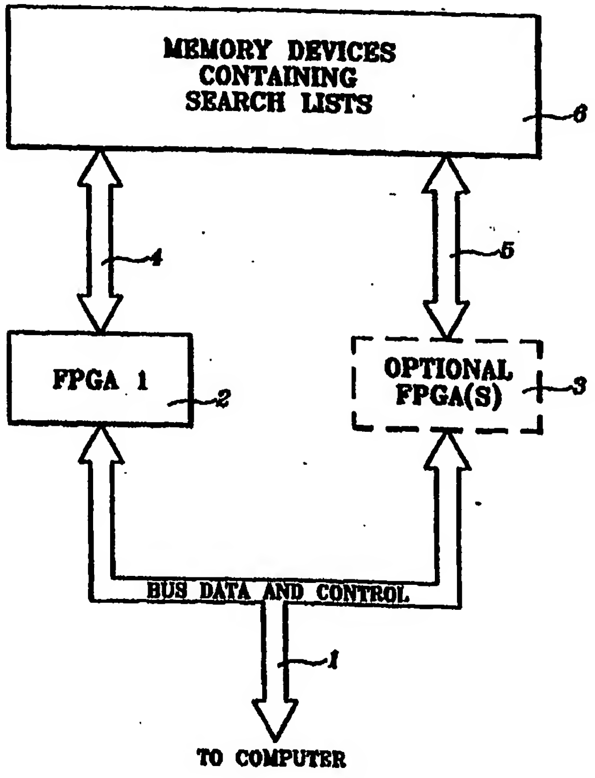
Published

Without international search report and to be republished upon receipt of that report.

(54) Title: MODULAR SYSTEM FOR ACCELERATING DATA SEARCHES AND DATA STREAM OPERATIONS

(57) Abstract

Using a modular reconfigurable Logic architecture coupled with a dense and flexible packaging scheme, it is possible to develop an engine with very high search speed and capable of complex search operations or data stream operations. This technology has great applicability in the areas of data mining, recognition of continuous speech, automated translation and image analysis/processing.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania
AM	Armenia
AT	Austria
AU	Australia
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CU	Cuba
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GH	Ghana
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IL	Israel
IS	Iceland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakstan
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
MK	The former Yugoslav Republic of Macedonia
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
MX	Mexico
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SZ	Swaziland
TD	Chad
TG	Togo
TJ	Tajikistan
TM	Turkmenistan
TR	Turkey
TT	Trinidad and Tobago
UA	Ukraine
UG	Uganda
US	United States of America
UZ	Uzbekistan
VN	Viet Nam
YU	Yugoslavia
ZW	Zimbabwe

MODULAR SYSTEM FOR ACCELERATING DATA SEARCHES AND DATA STREAM OPERATIONS

TECHNICAL FIELD OF THE INVENTION

This invention generally relates to integrated circuit computing devices and to computer system designs. More specifically, it relates to a combination of memory devices and Field-Programmable gate Arrays, together forming a Module which can be used to accelerate list processing functions such as database searches, speech recognition, speech or text translation, data stream transformation as in video or image editing, or routing of communications messages.

BACKGROUND OF THE INVENTION

Most computers use a simple architecture of a single memory subsystem and a single processor or set of processors accessing that memory. As a result, many systems are unable to perform so-called data-stream operations efficiently, and, are limited by the memory bandpass of the system in achieving total performance.

This limits the ability of the conventional computer to handle large data sets (data-streams) at an adequate level of performance. Such performance limitation prevents deployment of, for example, speech to text and automated translation systems.

To overcome this issue, and to provide the capability that demanding data-stream operations place on a system, it is necessary to: (a) increase memory bandwidth, (b) increase compute power by parallel processing, and (c) define a compute/comparison engine running at very high speed.

The invention described herein addresses each of these issues and achieves a dramatic performance boost for these type of operations. It provides a means to increase memory bandwidth by adding semi-autonomous Modules, it adds several layers of parallelism in computing, data transforming or comparison. The architecture is designed for the specific set of tasks required, but, since it is based on Reconfigurable Logic, the electronic circuits on which it is based can be rapidly modified at any given point in time to be optimal in configuration for the task at hand.

SUMMARY OF THE INVENTION

Using a combination of memory devices and field Programmable gate Arrays (FPGA), it is possible to build a modular system for accelerating data searches to much higher levels of performance than can bed realized with a simple computer system. This system achieves performance improvement in several ways.

First, the use of a combination of memory devices and FPGA's allows a much higher effective memory access rate than conventional computer architectures, with total memory bandpass increasing as each new module is added.

Second, because the architecture is independent of any computer structure the speed of access of each module to its memory component can be optimized to take advantage of special high-speed memory access modes such as fast page mode.

Third, the comparisons and other functions take place at hardware speeds, since the modular architect described herein does not require the structure of program steps typically seen in a conventional computer system.

Fourth, complex comparisons that involve logical or mathematical transforms of either the Search List data or the Search Target data can occur in a pipelined stream of hardware operations, permitting very sophisticated and complex operations, which, again, occur at hardware speeds.

The memory devices and FPGA's that make up a module can be packaged together in a variety of ways. Packaging choices include placing the elements on an adapter card that plugs into the computer bus, or into a special bus dedicated to the search functions. To achieve a dense and flexible packaging means, the combination of devices that makes up a module can be packaged onto a SIMM, DIMM or similar plug-in module. This permits the modules to be packed closely together, and allows the system designer choices as to whether the module is inserted into the sockets on the main processor board, or into sockets on a separate adapter card, where the constraints of the computer memory system can be ignored.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and for further advantages thereof, reference is now made to the following Description of the Preferred Embodiments taken in conjunction with the accompanying Drawings in which:

Figure 1 identified the basic structure of this invention, showing the connection of the various elements and optional elements and the function of the interconnections.

Figure 2 is an alternative method of connecting the elements together.

Figure 3 shows the functional content of the FPGA(s).

Figure 4 identified the incorporation of a processor or programmable controller element into the FPGA(s).

Figure 5 demonstrates how parallel function is achieved within a FPGA(s).

Figure 6 shows the preferred packaging scheme.

Figure 7 shows a scheme for connection of multiple Modules.

Figure 8 shows the connection of multiple Modules to operate on a large number of characters in parallel.

Figure 9 shows how multiple parallel comparisons are made using the same data lists.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The most basic embodiment of this invention is shown in Figure 1. Data and control (such as timing signals and address signals) are transferred from the computer on the Bus Data and Control lines 1 and into the FPGA 2 and/or additional optional FPGA's 3.

In the FPGA's 2,3, the data and control signals are modified to generate the Modified Data and Control signals 4 which are used to control the actions and contents of the memory Devices 6. Such modifications may include: 1) generating different address values than the one sent by the computer, 2) generating the required control and address values to permit reading data from the Memory Devices 5 to compare with values loaded into the FPGA(s) 2,3.

There are alternative methods of connecting the Memory Devices 5 to the FPGA(s) 2,3. Figure 2 shows one such alternative method, where the same Modified Data and Control 4 are shared by all the FPGA's 2,3, as opposed to the method shown in Fig. 1 where different Modified Data and Control 4,5 go to each FPGA 2,3. Such alternative methods are reconfigurable by connection of different logic in the FPGA(s) 2,3. This allows different operations to be performed in the several FPGA(s) 2,3 in the case of Figure 1, while the method of Figure 2 permits operation on the same or related data.

In Figure 3, the elements within the FPGA(s) 2,3 are detailed. Here is shown how data from the Memory Devices containing Search Lists 6 are moved into and from the FPGA(s) 2,3 with some combination of Transforms 7, Math Functions 8 and Comparators 9 being used to modify and/or examine the data. For clarity, only one of each such Transform 7, Math Function 8 or Comparators 9 is shown. A typical embodiment might have several of each, in any order, connected to operate consecutively on data. The control logic 10 manages the sequence of events inside FPGA(s) 2,3.

To effect a typical search, data constituting Search Lists are placed into the Memory Devices 6. Depending on application of the embodiment, this might be done by using rapidly reprogrammable Memory Devices, such as dynamic Random Access Memory (DRAM) or Static Random Access Memory (SRAM), semi-static memory devices that are typically programmed infrequently or only at the time of initial assembly of the embodiment, such as FLASH memory or electrically Erasable Programmable read-Only Memory (EEPROM) or one-time programmable Memory Devices such as Mask-Programmable Read-Only Memory (ROM).

Following the placement of the Search List data, the FPGA(s) 2,3 are re-programmed from an initial start-up state to be able to manipulate the Search List data now stored in the Memory Devices 6. Such manipulations are effected by placing the functional elements, Transforms 7, Math/Logic Functions 8 and Comparators 9 in any sequence or quantity to act upon selected data elements of the Search List data.

A data item (Search Target) to be compared against the Search List is placed into FPGA's 2,3. Data from the Search List are then moved, data item by data item, into the FPGA(s) 2,3, where the instantiated Transforms 7 and Math/Logic Functions 8 operate on said Search List data item, following which said modified data item is compared with the Search Target inside Comparator 9. If a match is found between the Search Target and the Search List data item, the Control Logic 10 then informs the computer that a match has been found. Said Control Logic may be programmed to continue on for additional matches to the same Search target data, or re-loaded with new Search Target data, and the Search List and FPGA contents may be changed at any time as required to optimize performance.

Figure 4 extends the concept described above to allow a programmable controller or processor 11 to be instantiated into the FPGA. This permits much greater flexibility in operation, since the sequence of hardware events, and the interaction of the module(s) with a host computer are capable of being modified.

Figure 5 shows an extension of the embodiment where multiple search operations occur in parallel. This is realized by instantiating sets of the various Transforms 7, Math/Logic Functions 8 and Comparators 9 into FPGA(s) 1 (etc.) and loading either the same or different Search Target data elements to correspond with each such set, which may contain different sizes and types of transforms 7, functions 8 and comparators 9. The operation in such multiple search mode follows the sequence above for a single search path, with the set of Search Target data items being compared with either the same Search List data items, as (optionally) modified by the (possibly different) set of Transforms 7 that are applied in each search path, or with different Search List data items, similarly modified.

The preferred packaging scheme (Figure 6) for the Modules is the SIMM. In this means, Memory Devices 6 and FPGA's 1,2 are mounted on one of several industry-standard form-factor boards to make a Module. This permits a very dense package, taking up a small physical space, and advantageously is supported by many computer systems. Alternative packaging schemes include the industry standard PCM-CIA bus card, the DIMM card, the small footprint PCI card and many other standard form factors.

In a typical application, several Modules 16 will be mounted together to achieve modular increments of power. Figure 7 shows such a configuration. Note that each Module shares the Data and Control signals to the computer. This permits each Module 16 to be loaded with Search Data, Search Target and control information, and to communicate with the computer, while allowing the autonomous parallel operation of the Modules 16 during the searching or modifying of data.

The Modules 16 can also be connected in such a way as to communicate with each other. This permits comparison of very wide data elements, which might be useful in image or speech processing, for example. Figure 8 shows a means where this might be achieved by sharing the computer Data and Control Box 1, which is connected to all of the Modules, as an intercommunication path 12 between each Module. Determination of the success or otherwise of the search or data modification operations can be realized by either the computer system or a specially programmed Module 13.

Another method of using the Module architecture, shown in Figure 9, is to build several parallel search or transform paths in each Module. This can be done within a single FPGA, as shown, or within multiple FPGA's mounted on the same module and sharing the same data. This method has the benefit that different transforms, mathematical operations or comparison methods can be deployed in parallel, to act on the same data, or, if appropriate, different data, as required. This allows, in some circumstances, for a large multiplication of performance of the modular system.

CLAIMS:

1. A data processing module adapted to be connected to a computer for use with a computer, the computer including a memory for storing data, the module comprising:

a module memory for storing data; and

a programmable logic device connected to said module memory and adapted to be connected to the computer for receiving data stored in said module memory and the computer memory for processing data.

2. The module of Claim 1 wherein said programmable logic device includes a comparator for determining whether data stored in the computer memory is stored in said module memory.

3. The module of Claim 1 wherein said programmable logic device is programmable by data stored in said module memory for processing data stored in said module memory.

4. The module of Claim 1 wherein said module memory includes a random access memory device.

5. The module of Claim 1 wherein said module memory and said programmable logic device are mounted on a single in-line memory module having terminals for connection to the computer.

6, A data processing system for use with a computer, the computer including a memory for storing data, the system comprising:

a plurality of data processing modules, adapted to be connected to the computer, each of said modules including:

a module memory for storing data; and

a programmable logic device connected to said module memory and adapted to be connected to the computer for receiving data stored in said module memory and the computer memory; and

such that said plurality of data processing modules simultaneously process data stored in each of said module memories and the computer memory.

7. The system of Claim 6 wherein said programmable logic devices include a comparator for determining whether data stored in the computer memory is stored in said module memories.

8. The system of Claim 6 and further including:

means for transferring data between said plurality of data processing modules.

9. The system of Claim 6 wherein ones of said programmable logic devices perform comparisons on said data stored in said module memories.

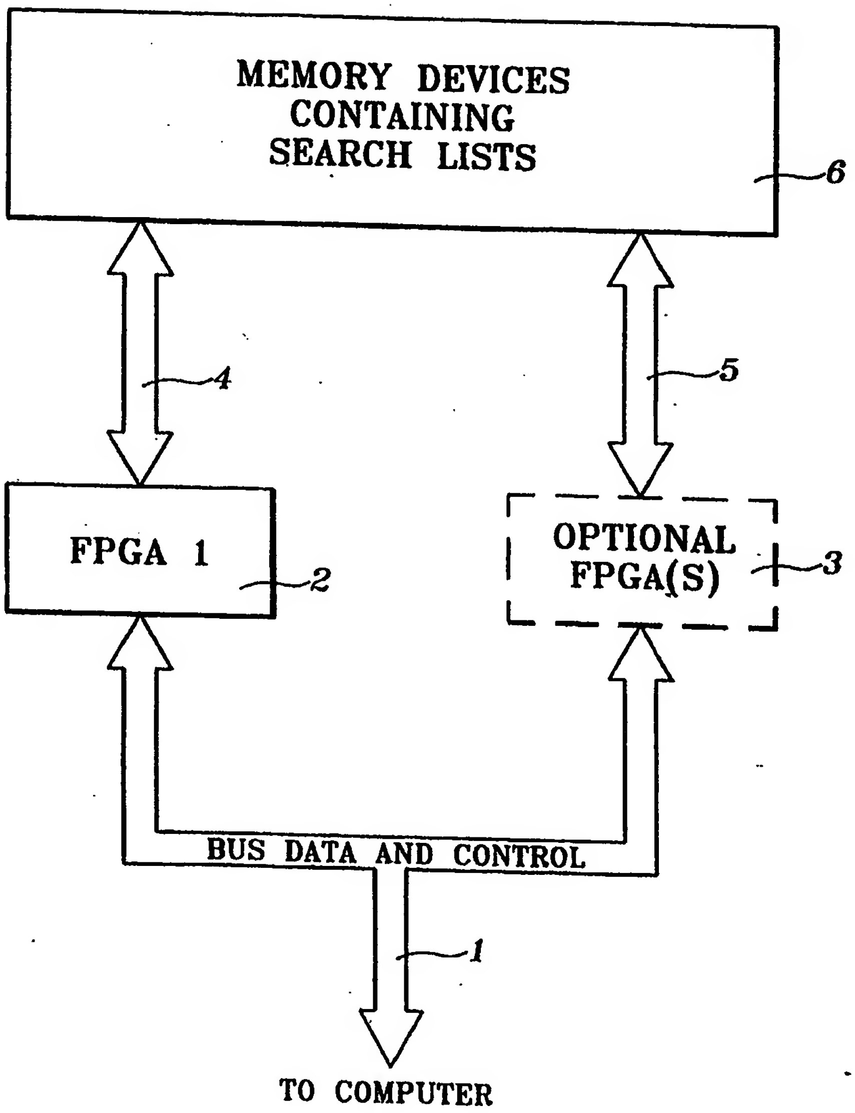

MEMORY DEVICES
CONTAINING
SEARCH LISTS
6
4
5
FPGA 1
2
OPTIONAL
FPGA(S)
3
BUS DATA AND CONTROL
1
TO COMPUTER

FIG. 1

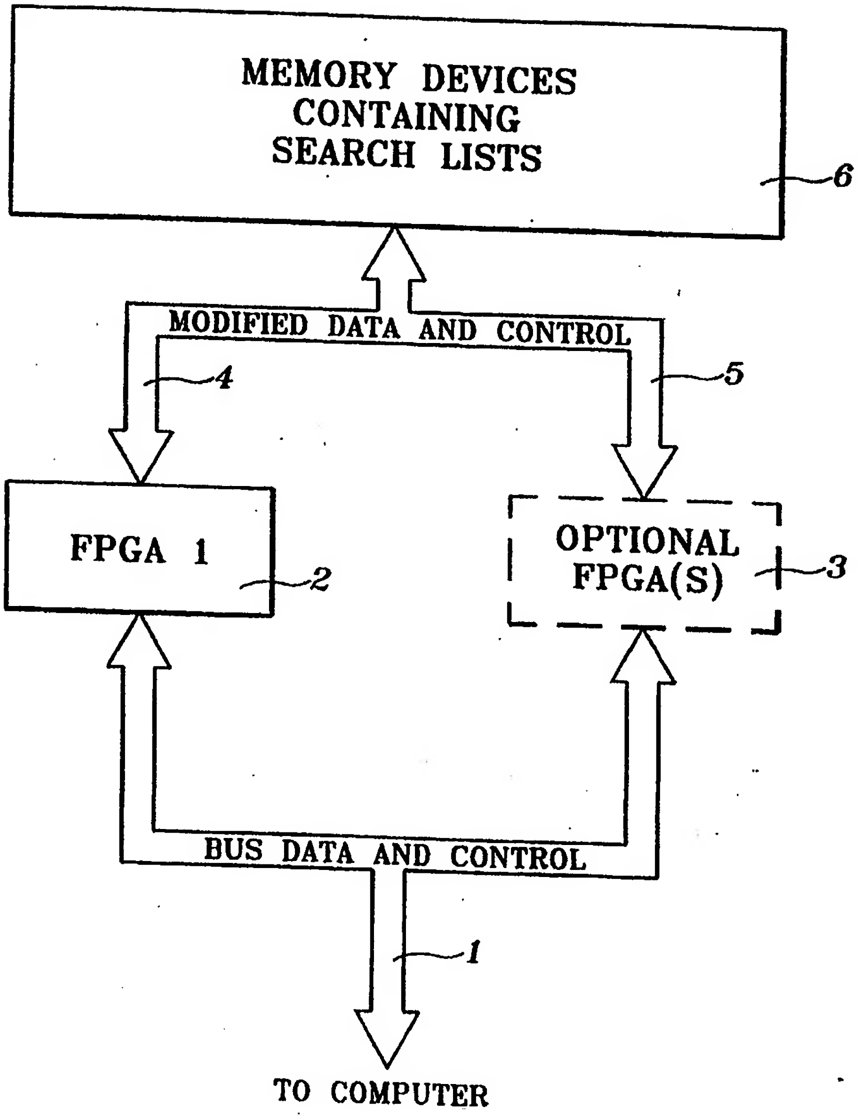

MEMORY DEVICES CONTAINING SEARCH LISTS
6
MODIFIED DATA AND CONTROL
4
5
FPGA 1
2
OPTIONAL FPGA(S)
3
BUS DATA AND CONTROL
1
TO COMPUTER

FIG. 2

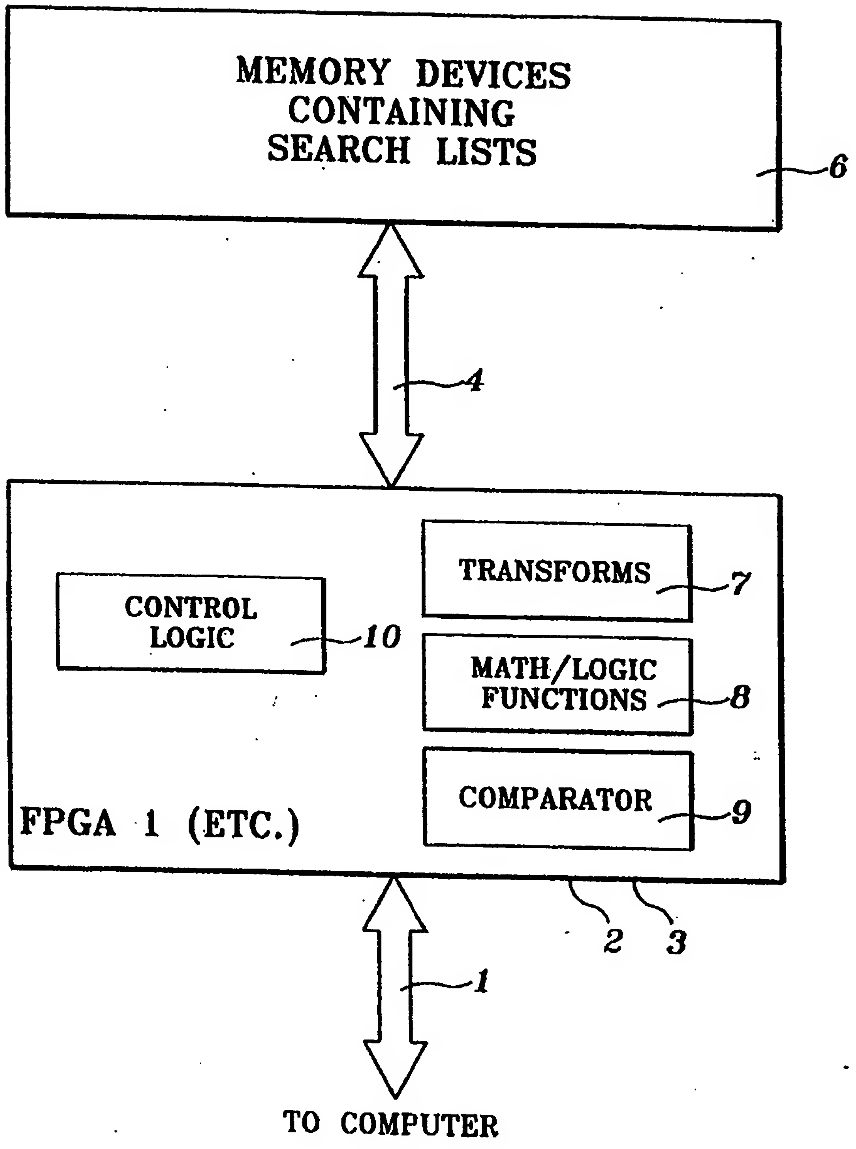

MEMORY DEVICES CONTAINING SEARCH LISTS
6
4
CONTROL LOGIC
10
TRANSFORMS
7
MATH/LOGIC FUNCTIONS
8
COMPARATOR
9
FPGA 1 (ETC.)
2
3
1
TO COMPUTER

FIG. 3

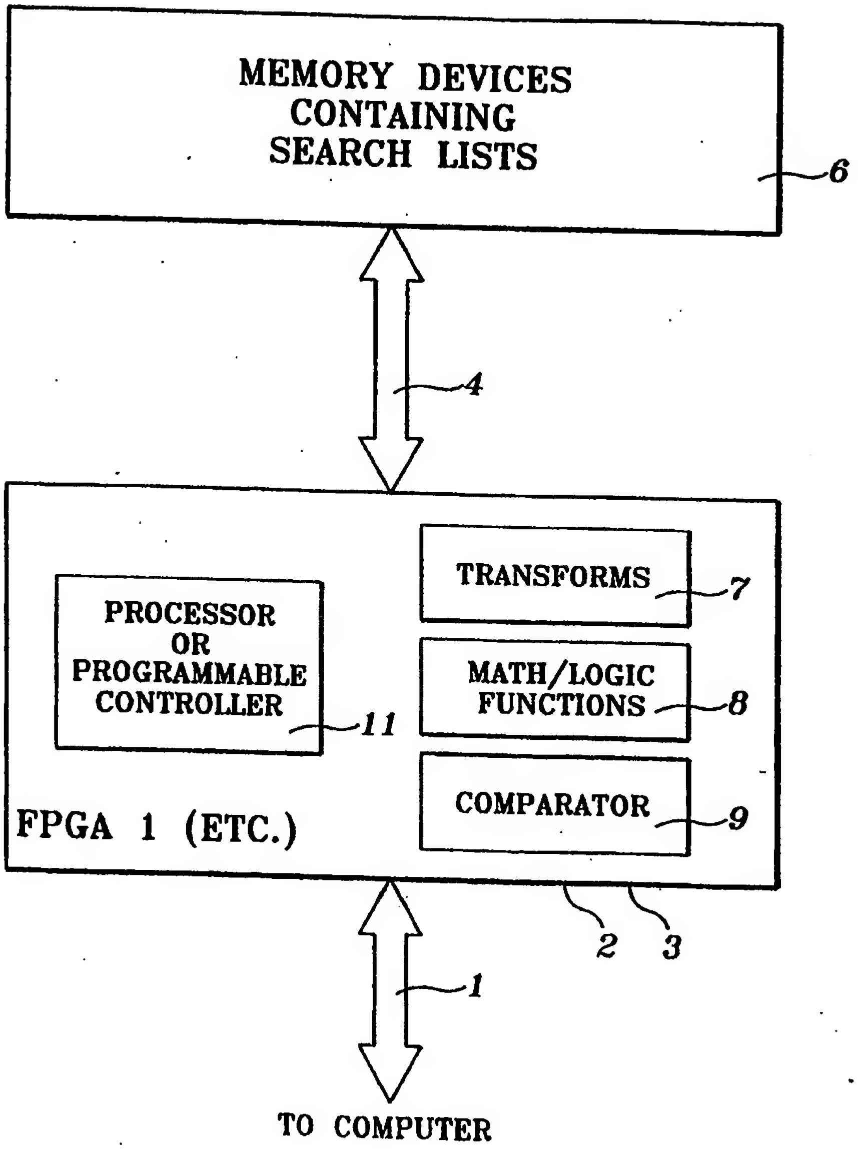

MEMORY DEVICES CONTAINING SEARCH LISTS
6
4
TRANSFORMS
7
PROCESSOR OR PROGRAMMABLE CONTROLLER
MATH/LOGIC FUNCTIONS
8
11
COMPARATOR
9
FPGA 1 (ETC.)
2
3
1
TO COMPUTER

FIG. 4

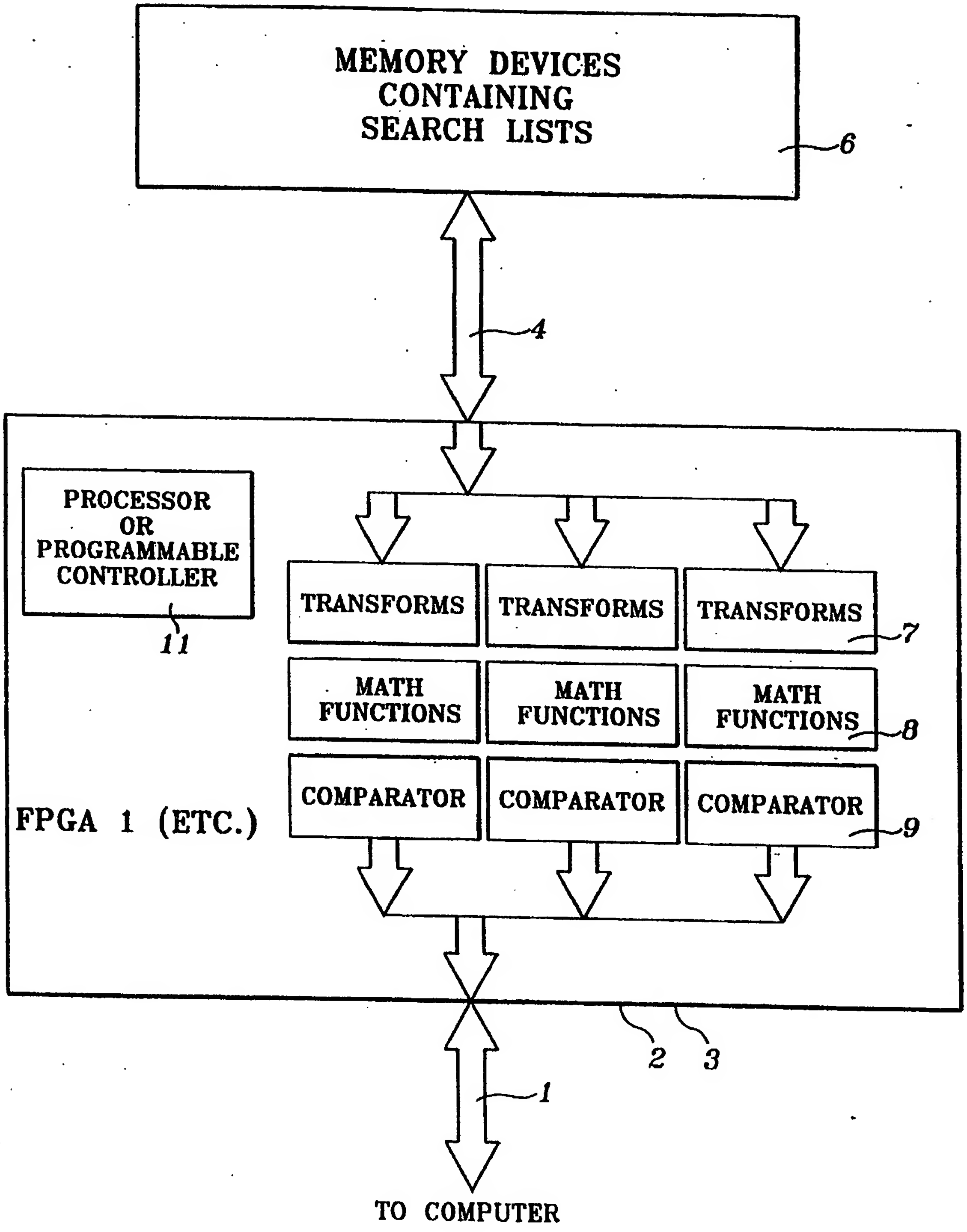

MEMORY DEVICES CONTAINING SEARCH LISTS
6
4
PROCESSOR OR PROGRAMMABLE CONTROLLER
11
TRANSFORMS
TRANSFORMS
TRANSFORMS
7
MATH FUNCTIONS
MATH FUNCTIONS
MATH FUNCTIONS
8
COMPARATOR
COMPARATOR
COMPARATOR
9
FPGA 1 (ETC.)
2
3
1
TO COMPUTER

FIG. 5

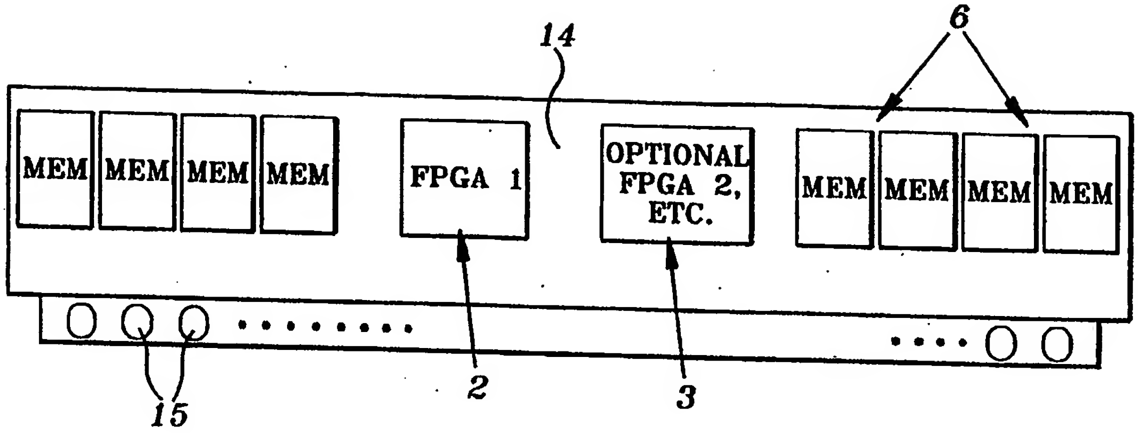

14
6
MEM
MEM
MEM
MEM
FPGA 1
OPTIONAL FPGA 2, ETC.
MEM
MEM
MEM
MEM
15
2
3

FIG. 6

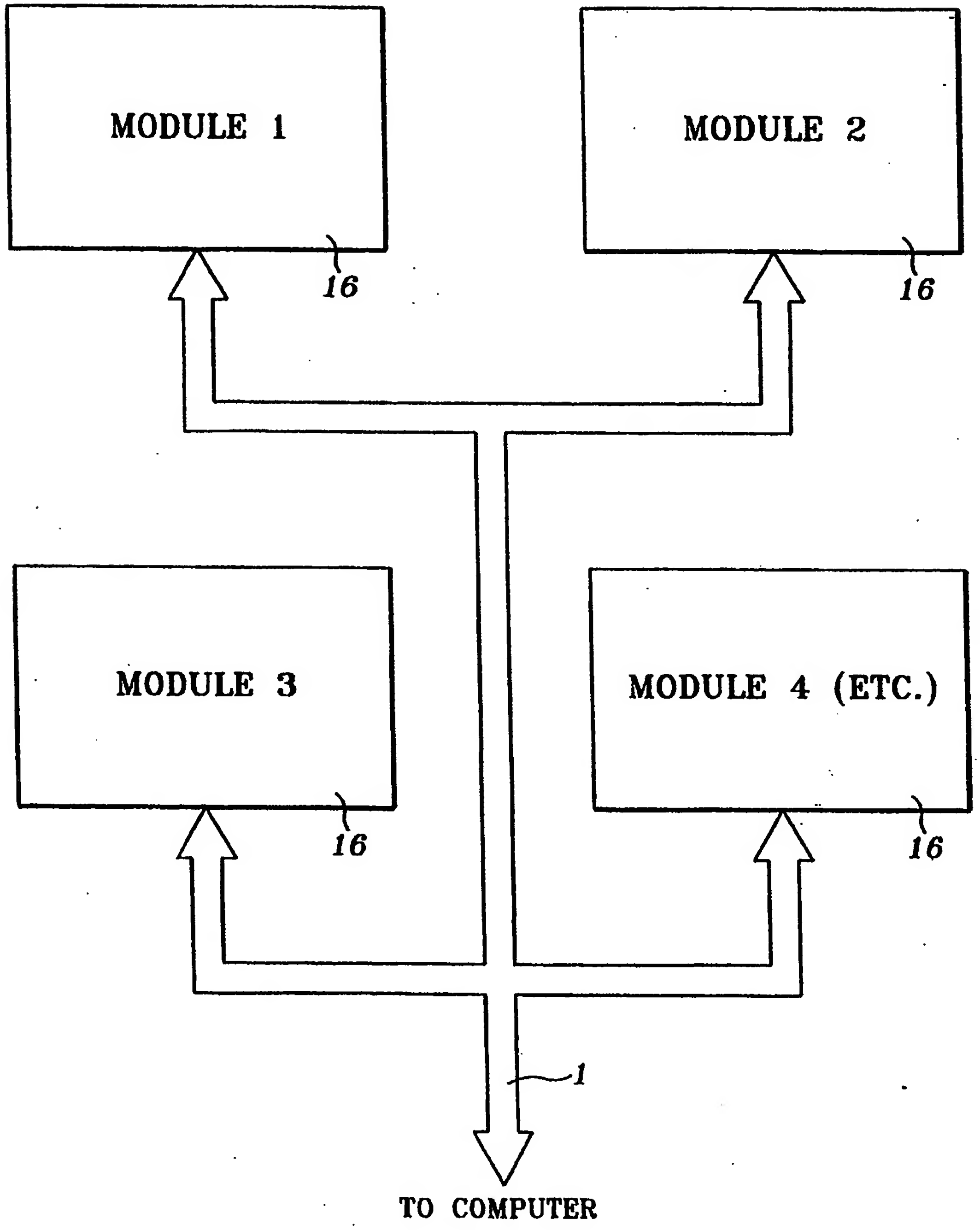

MODULE 1
16
MODULE 2
16
MODULE 3
16
MODULE 4 (ETC.)
16
1
TO COMPUTER

FIG. 7

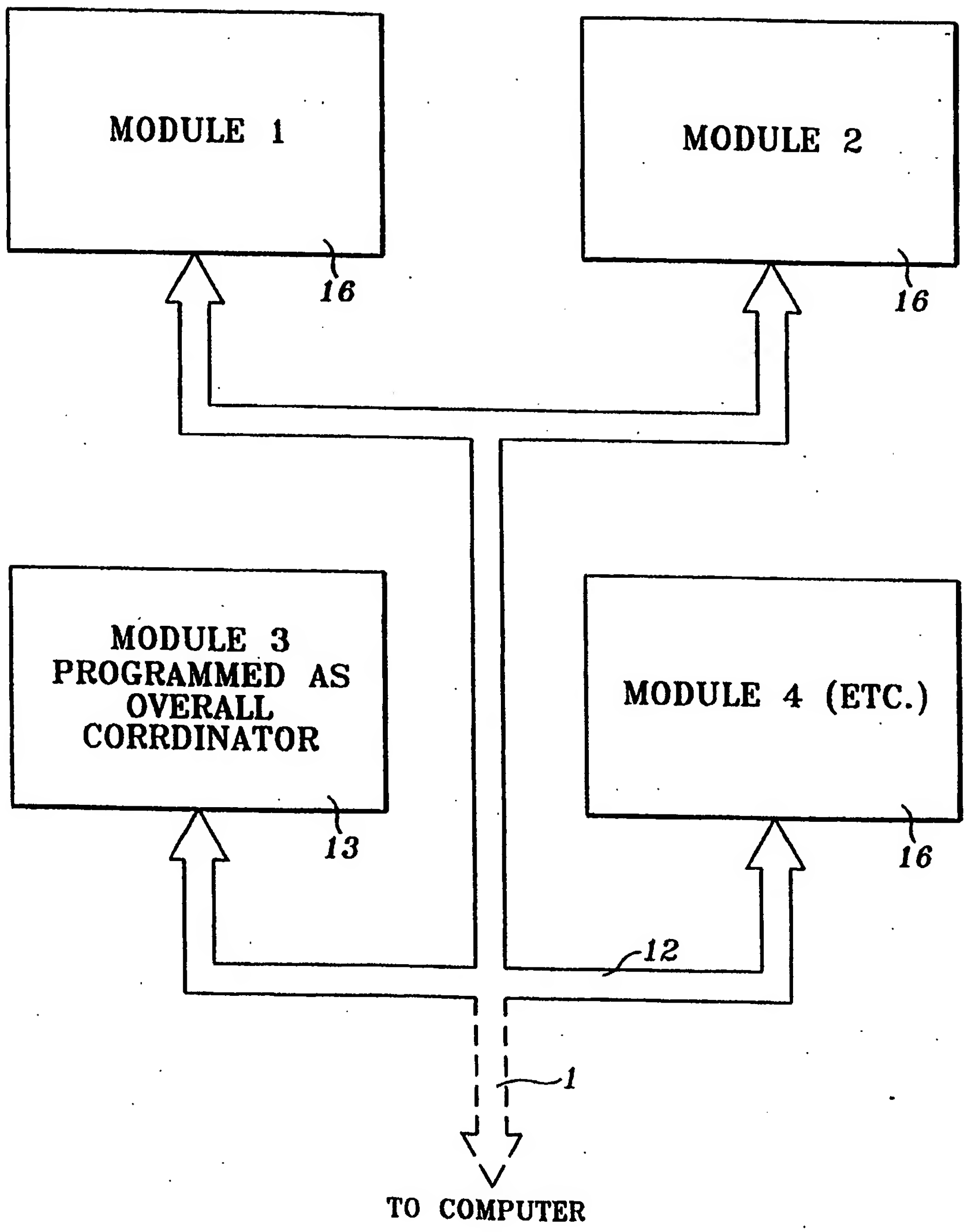

MODULE 1
16
MODULE 2
16
MODULE 3 PROGRAMMED AS OVERALL CORRDINATOR
13
MODULE 4 (ETC.)
16
12
1
TO COMPUTER

FIG. 8

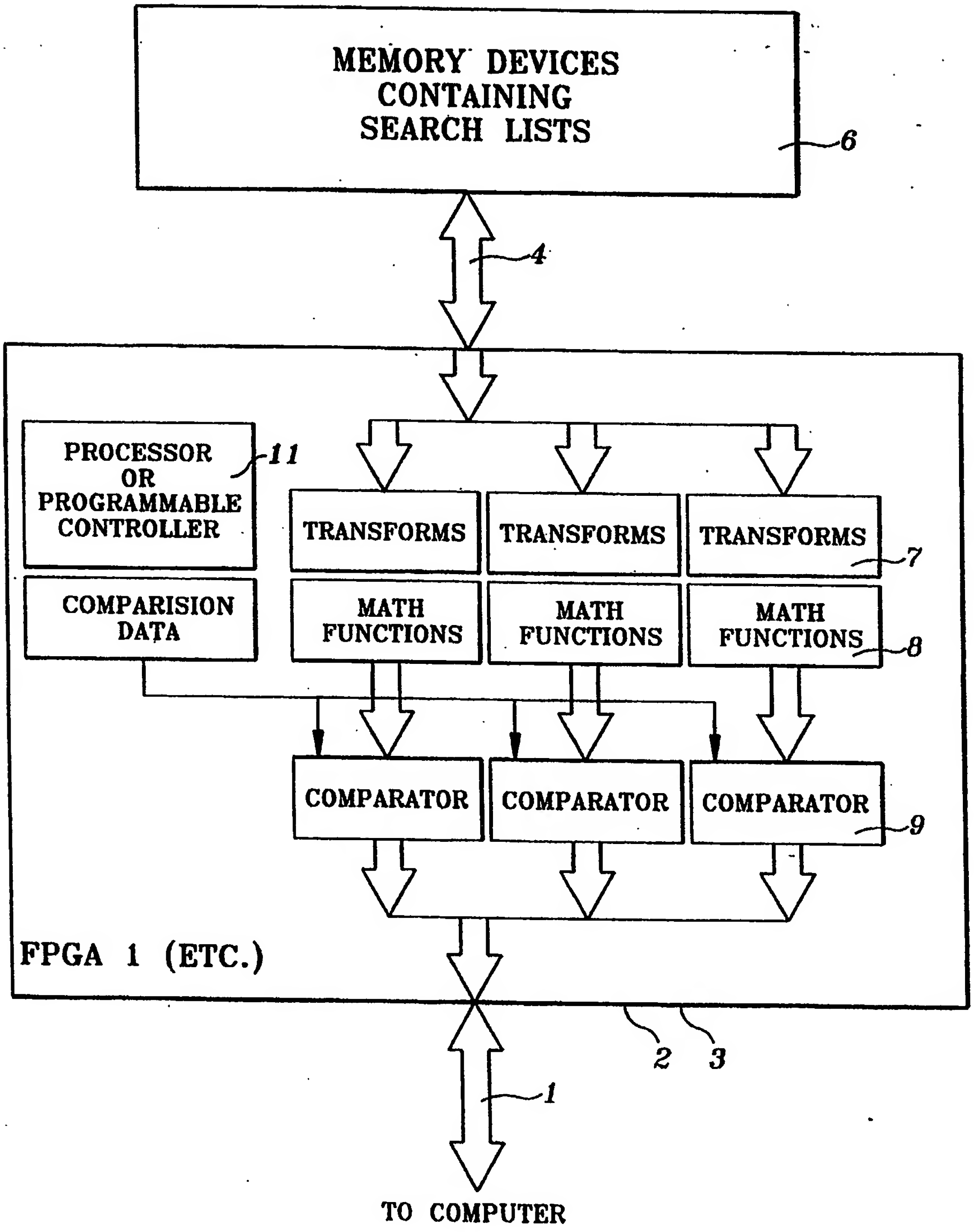

MEMORY DEVICES CONTAINING SEARCH LISTS
6
4
PROCESSOR OR PROGRAMMABLE CONTROLLER
11
TRANSFORMS
TRANSFORMS
TRANSFORMS
7
COMPARISION DATA
MATH FUNCTIONS
MATH FUNCTIONS
MATH FUNCTIONS
8
COMPARATOR
COMPARATOR
COMPARATOR
9
FPGA 1 (ETC.)
2
3
1
TO COMPUTER

FIG. 9

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 April 2002 (04.04.2002)

PCT

(10) International Publication Number
WO 02/27499 A2

(51) International Patent Classification⁷: G06F 12/10

(21) International Application Number: PCT/US01/30362

(22) International Filing Date:
26 September 2001 (26.09.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/676,844 29 September 2000 (29.09.2000) US

(71) Applicant *(for all designated States except US)*: INTEL CORPORATION [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: WHITE, Bryan [US/US]; 6051 West Shannon Street, Chandler, AZ 85226 (US).

(74) Agents: HARRIS, Scott, C. et al.; Fish & Richardson, 4350 La Jolla Village Drive, San Diego, CA 92122 (US).

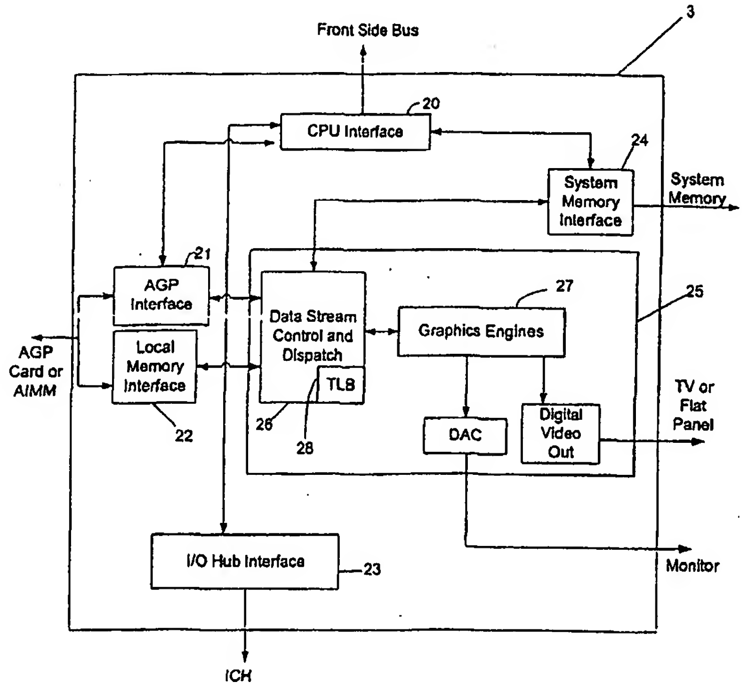
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SHARED TRANSLATION ADDRESS CACHING



(57) Abstract: A memory controller hub includes a graphics subsystem adapted to perform graphics operations on data and a cache adapted to store of locations in physical memory available to the graphics subsystem for storing graphics data and available to a graphics controller coupled to the memory controller hub to store graphics data.

SHARED TRANSLATION ADDRESS CACHING

BACKGROUND

The invention relates to caching in memory controller hubs.

Microcomputer systems generally include one or more memory controller hubs that control and coordinate the transfer of data between the computer's system memory, central processing unit (CPU), and peripheral devices. Graphics applications may be supported by peripheral devices known as graphics controllers that require a memory controller hub to transfer data between the graphics controller, the system memory, and the CPU.

A design concern associated with microcomputer systems is the quality of two-dimensional (2D), three-dimensional (3D), and video image (collectively referred to below as "graphics") processing. High-performance graphics processing requires processor-intensive calculations and the fast manipulation of large quantities of data. Several designs have been implemented to achieve high-performance graphics processing while also reducing the cost of the complete system and allowing for upgrades to the computer system's capability.

A computer system may include a graphics controller coupled to local memory for storing graphics data, so that the amount of data that must be transferred between the graphics controller and the system memory and/or the CPU is reduced. Increasing the amount of local memory available to the graphics controller improves graphics performance, but also increases the cost of the computer system, because local graphics memory is relatively expensive. Less local memory is required to achieve the same graphics performance, however, if a dedicated bus, *e.g.*, an Accelerated Graphics Port (AGP), is used to couple the controller to the memory controller hub. An AGP allows the controller to treat portions of system memory as dedicated local graphics memory, which reduces the amount of local memory required and lowers overall system costs.

Computer system costs also may be reduced by eliminating the peripheral graphics controller and integrating its functionality into the memory controller hub. In such a configuration the memory controller hub is better described as a graphics/memory controller hub, since it performs graphics processing functions in addition to memory control and transfer functions. Additionally, it includes one or more output ports to send graphics signals to external devices, such as cathode ray tubes (CRTs) and flat-panel monitors. A graphics/memory controller hub may be coupled to local memory for storing graphics data.

BRIEF DESCRIPTION OF DRAWINGS

Figure 1 is a schematic block diagram of a computer system.

Figure 2 is a schematic block diagram of a graphics memory controller hub.

Figure 3 and 4 is a schematic block diagram of accelerated graphics port (AGP) functionality of a graphics memory controller hub.

DETAILED DESCRIPTION

In a computer system, a memory controller hub may be integrated with an internal graphics controller and may interface with an external graphics device through an AGP port. Because the memory controller hub controls both graphics and memory functions it is referred to as a graphics/memory controller hub (GMCH). The GMCH provides both internal graphics processing and scalable graphics performance through an AGP interface.

The GMCH may be used in one of two mutually exclusive modes: AGP mode, in which case the GMCH uses its capability to interface with an external graphics controller and its internal graphics functionality is disabled; or Gfx mode, in which case the GMCH uses its internal graphics capability, and its ability to interface with an external graphics controller is disabled. In Gfx mode the GMCH can still interface with a local memory module through the AGP port to provide additional graphics memory for use by the internal graphics. Whether the GMCH operates in AGP mode or Gfx mode can be determined automatically and set during the start-up sequence of the computer.

Figure 1 illustrates an exemplary computer system 1 in which the GMCH can be implemented. The computer system 1 includes a microprocessor (for example, CPU) 2 coupled to a GMCH 3, which contains a system memory controller hub. GMCH 3 may also be referred to as a "chipset" or "core logic." GMCH 3 provides an interface between CPU 2 and system memory 4, and between CPU 2 and a bus, for example, a peripheral component interconnect (PCI) or Hublink™ bus 5. Various input/output (I/O) devices 6 are coupled to PCI bus 5, which is coupled to GMCH 3 via input/output controller hub (ICH) 11. Computer system 1 may also include a graphics device 7, which may be a graphics controller coupled to local memory 8, or which may be an AGP Inline Memory Module (AIMM) that provides external local memory for the internal graphics functionality of GMCH 3. A shared AGP/local memory interface 9 provides a dedicated interface bus between GMCH 3 and graphics device 7. Graphics and video signals may be sent to a display device 10 from graphics device 7 if one is present in the computer system, or may be sent to display device 10 from GMCH 3 if graphics device 7 is absent.

Figure 2 illustrates other details of GMCH 3, including a CPU interface 20 coupled to an AGP interface 21, a local memory interface 22, an input/output (I/O) hub interface 23, and a system memory interface 24. Graphics functions can be performed by internal graphics components 25, which include a data stream and dispatch controller 26 to manage the flow of data and various graphics engines 27 to perform graphics operations on data.

Referring to Figure 3 and 4, AGP transactions are run in a split transaction fashion in which a request for data transfer to or from system memory 4 is disconnected in time from the data transfer itself. An AGP compliant graphics device (bus master) 7a initiates a transaction with an access request. The AGP interface 21 responds to the request by directing the corresponding data transfer at a later time, which permits the AGP graphics device 7a to pipeline several access requests while waiting for data transfers to occur. As a result of pipelining, several read and/or write access requests may be simultaneously outstanding in request queues 100. Access requests can either be pipelined across an address/data bus (AD bus) 105, 107 of AGP 9 or transferred through sideband address lines 107 of AGP 9 and received by request queue 100.

Scheduler 102 processes the access requests in request queue 100. Read data are obtained from system memory 4 and are returned at the initiative of scheduler 102 through read data return queue 104 and across AD bus 105 of the AGP 9. Write data are provided by AGP compliant graphics controller 7 at the direction of scheduler 102 when space is available in the write data queue 108. Thus, AGP transactions generally include interleaved access requests and data transfers.

Graphics data may be stored in system memory 4 when GMCH 3 operates in AGP mode in conjunction with an external AGP compliant graphics controller 7a, or when GMCH 3 operates in Gfx mode using its internal graphics functionality. When using system memory 4 to store graphics data, GMCH 3 uses a virtual memory addressing concept for accessing graphics data. In AGP mode, a 32 MB or 64 MB graphics aperture is defined through which addresses in the physical system memory 4 can be accessed by graphics controller 7a. The graphics aperture appears to graphics controller 7a as a 32 MB or 64 MB contiguous block of linear memory, although the addresses in physical system memory allocated for use by AGP graphics controller 7a are not contiguous. The contiguous block of memory addresses in the graphics aperture permits graphics controller 7a to quickly access large data structures, such as texture bitmaps (typically 1 KB to 128 KB), as single entities in virtual memory.

Access requests from graphics controller 7a address virtual memory within the aperture range, and then GMCH 3 forwards access requests within the aperture to physical system memory 4. The originally-issued addresses sent from graphics controller 7a are translated within data stream controller 26 using a Graphics Address Remapping Table (GART). The GART is a table that matches virtual memory address in the aperture range with corresponding physical memory addresses. The GART is stored in system memory in a location known to GMCH 3 because its location is stored in a register within GMCH 3. Addresses are mapped from the graphics aperture into system memory in 4 KB pages, and each entry of the GART translates one 4 KB page. Thus, when an access request is received from graphics controller 7a in the graphics aperture, the request is momentarily stalled while the appropriate GART entry is fetched from system memory 4. The address of the access request within the graphics aperture is translated using the fetched translation table entry, and the request is forwarded to the physical address in system memory 4 identified by the fetched GART entry.

To speed up memory access requests from AGP graphics controller 7a to system memory 4, GMCH 3 provides a GART entry cache 28 for locally storing up to four entries from the GART. GART entry cache 28 may also be known as a translation lookaside buffer (TLB). When a GART entry is first retrieved from the GART in system memory 4 to translate a virtual address into a physical address, the entry may be stored in the TLB 28 residing in data stream controller 26. The next time an address request from graphics controller 7a needs to use the same GART entry, the entry can be retrieved from the local TLB 28 rather than from the GART in distant system memory 4. Since GART entries may be stored in TLB 28 and each GART entry provides access to a 4KB page of memory addresses, up to 16 KB of access requests from graphics controller 7a may be translated using the GART entries stored locally in TLB 28 before a new GART entry must be retrieved from system memory 4. If data stream controller 26 needs to use a GART entry that is not stored locally in TLB 28, the necessary entry may be retrieved from system memory 4 and then stored in TLB 28 for future use, thereby replacing an entry previously stored in TLB 28.

Referring again to Figure 2, in Gfx mode, the internal graphics engines 27 of GMCH 3 define a 64 MB logical address space through which addresses in the physical system memory 4 or AIMM can be accessed by internal graphics engines 27. The logical address space appears to graphics controller 7a as a 32 MB or 64 MB contiguous block of linear memory, although the addresses in physical system memory 4 or AIMM allocated for use by internal graphics engines 27 are not contiguous. Like the graphics aperture used in AGP mode, the contiguous block of memory addresses in the logical address space permits internal graphics engines 27 to access large data structures quickly as single entities in virtual memory.

Access requests from internal graphics engines 27 are translated within data stream controller 26 using a Graphics Translation Table (GTT), which is stored in system memory in a location stored by GMCH 3 in a register within GMCH 3. Addresses within the logical address space are mapped into system memory or AIMM in 4KB pages, and each entry of the GTT translates one 4KB page. GTT entries additionally determine whether access requests are mapped to system memory 4 or AIMM memory, if an AIMM card is present. The same TLB 28 used in GMCH 3 to cache GART entries may be used to store up to four entries locally from the GTT in order to speed access to physical memory. Since the number of GART entries or GTT entries that may be stored in TLB 28 is limited by the physical die area size of the TLB, using the same TLB to store GART entries in AGP mode and to store GTT entries in Gfx mode effectively doubles the number of GART or GTT entries that may be stored in TLB compared to the number that could be stored if separate TLBs were used for GART and GTT entries. Additionally, using the same TLB to store GART entries in AGP mode and to store GTT entries in Gfx mode simplifies the internal logic of GMCH 3 because a single logic serves both functions of the TLB.

Other embodiments are within the scope of the following claims.

What is claimed is:

1. A memory controller hub comprising:

a graphics subsystem adapted to perform graphics operations on data; and

a cache adapted to store addresses of locations in physical memory available to the graphics subsystem for storing graphics data and available to a graphics controller coupled to the memory controller hub to store graphics data.

2. The memory controller hub of claim 1 further including a dedicated bus interface coupling the graphics controller to the memory controller hub.

3. The memory controller hub of claim 2 wherein the dedicated bus interface includes an accelerated graphics port (AGP).

4. The memory controller hub of claim 1 configured to provide a block of linear, virtual memory addresses for use by the graphics subsystem, wherein the cache is adapted to store addresses of locations in physical memory that correspond to addresses within the block of linear, virtual memory addresses.

5. The memory controller hub of claim 1 configured to provide a block of linear, virtual memory addresses for use by the graphics controller, wherein the cache is adapted to store addresses of locations in physical memory that correspond to addresses within the block of linear, virtual memory addresses.

6. The memory controller hub of claim 1 configured to provide a first block of linear, virtual memory addresses for use by the graphics controller and adapted to provide a second block of linear, virtual memory addresses for use by the graphics subsystem, wherein the cache is adapted to store addresses of locations in physical memory that correspond to addresses within the first block of linear, virtual memory addresses and to store addresses of locations in physical memory that correspond to addresses within the second block of linear, virtual memory addresses.

7. A computer system comprising:

a CPU;

a display device;

a system memory adapted to store video data and non-video data; and

a memory controller hub coupled to the CPU and coupled to the system memory, the memory controller hub comprising:

a graphics subsystem configured to perform graphics operations on graphics data; and

a cache adapted to store addresses of locations in physical memory available to the graphics subsystem for storing graphics data and that available to a graphics controller coupled to the memory controller hub to store graphics data.

8. The computer system of claim 7 further including a dedicated bus interface coupling the graphics controller to the memory controller hub.

9. The computer system of claim 8 wherein the dedicated bus interface includes an accelerated graphics port (AGP).

10. The computer system of claim 7 wherein the memory controller hub is configured to provide a block of linear, virtual memory addresses for use by the graphics subsystem; and

wherein the cache is adapted to store addresses of locations in physical memory that correspond to addresses within the block of linear, virtual memory addresses.

11. The computer system of claim 7 wherein the memory controller hub is configured to provide a block of linear, virtual memory addresses for use by the graphics controller; and

wherein the cache is adapted to store addresses of locations in physical memory that correspond to addresses within the block of linear, virtual memory addresses.

12. The computer system of claim 7 wherein the memory controller hub is configured to provide a first block of linear, virtual memory addresses for use by the graphics controller and is adapted to provide a second block of linear, virtual memory addresses for use by the graphics subsystem; and

wherein the cache is adapted to store addresses of locations in physical memory that correspond to addresses within the first block of linear, virtual memory addresses and is adapted to store addresses of locations in physical memory that correspond to addresses within the second block of linear, virtual memory addresses.

13. A method of storing addresses of locations in physical in a memory controller hub cache wherein the locations in physical memory are available to either a graphics controller coupled to the memory controller hub or are available to a graphics subsystem of the memory controller hub.

14. The method of claim 13 further comprising:

providing a block of linear, virtual memory addresses the memory controller hub for use by the graphics subsystem; and storing in the cache addresses of locations in physical memory that correspond to addresses within the block of linear, virtual memory addresses.

15. The method of claim 13 further comprising:

providing a block of linear, virtual memory addresses in the memory controller hub for use by the graphics controller; and

storing in the cache addresses of locations in physical memory that correspond to addresses within the block of linear, virtual memory addresses.

16. The method of claim 13 further comprising:

providing a block of linear, virtual memory address the memory controller hub for use by the graphics controller, and storing in the cache addresses of locations in physical memory that correspond to addresses within the block of linear, virtual memory addresses; or

providing a block of linear, virtual memory address the memory controller hub for use by the graphics subsystem, and storing in the cache addresses of locations in physical memory that correspond to addresses within the block of linear, virtual memory addresses.

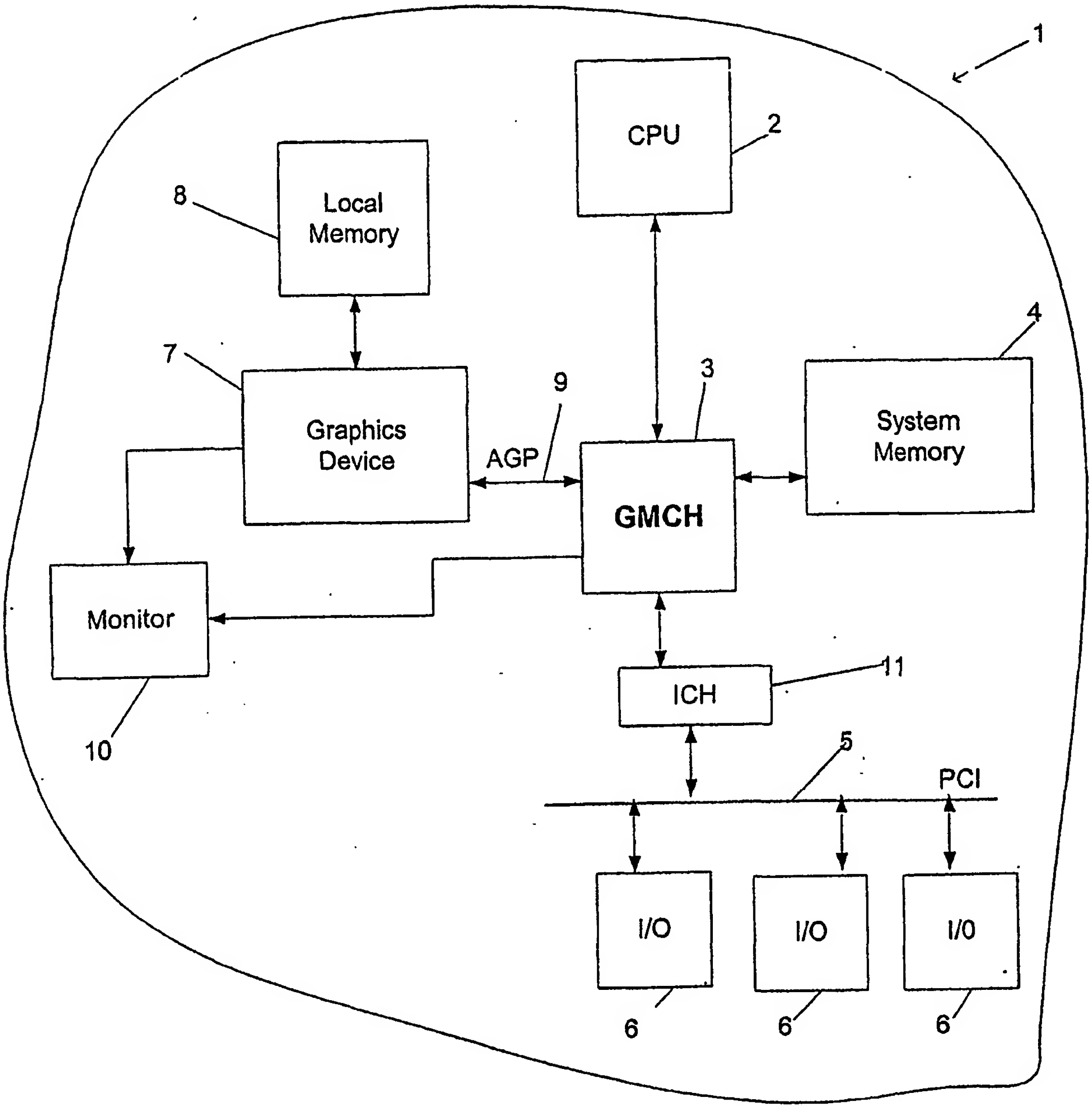

1
CPU
2
Local Memory
8
7
Graphics Device
9
3
4
System Memory
AGP
GMCH
Monitor
10
ICH
11
5
PCI
I/O
I/O
I/0
6
6
6

Figure 1

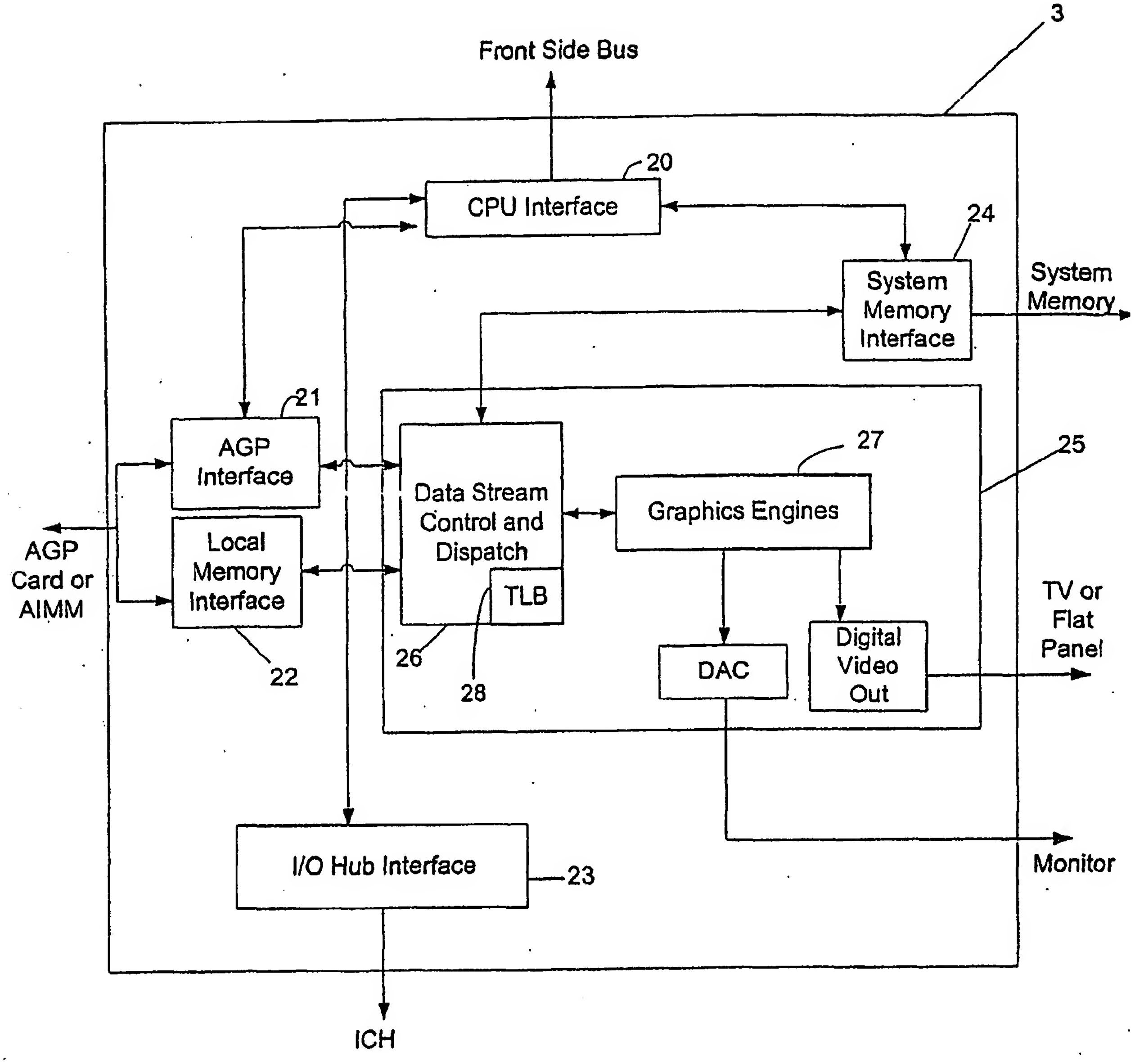

3
Front Side Bus
20
CPU Interface
24
System Memory Interface
System Memory
21
AGP Interface
Local Memory Interface
22
AGP Card or AIMM
Data Stream Control and Dispatch
TLB
26
28
27
Graphics Engines
25
DAC
Digital Video Out
TV or Flat Panel
I/O Hub Interface
23
Monitor
ICH

Figure 2

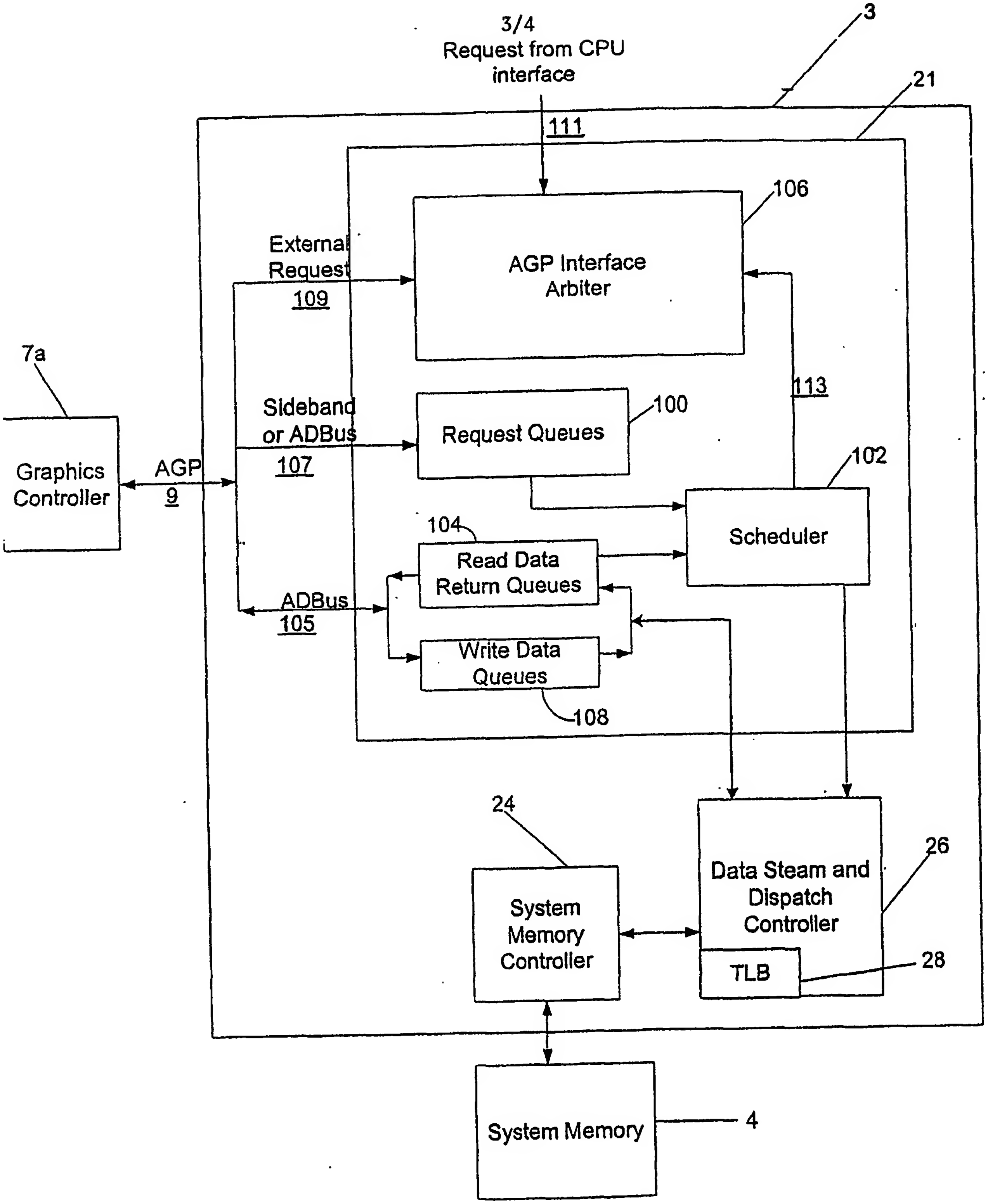

3/4
Request from CPU interface
3
21
111
106
AGP Interface Arbiter
External Request
109
Sideband or ADBus
107
Request Queues
100
113
102
Scheduler
7a
Graphics Controller
AGP
9
104
Read Data Return Queues
ADBus
105
Write Data Queues
108
24
System Memory Controller
Data Steam and Dispatch Controller
26
TLB
28
System Memory
4

Figure 3

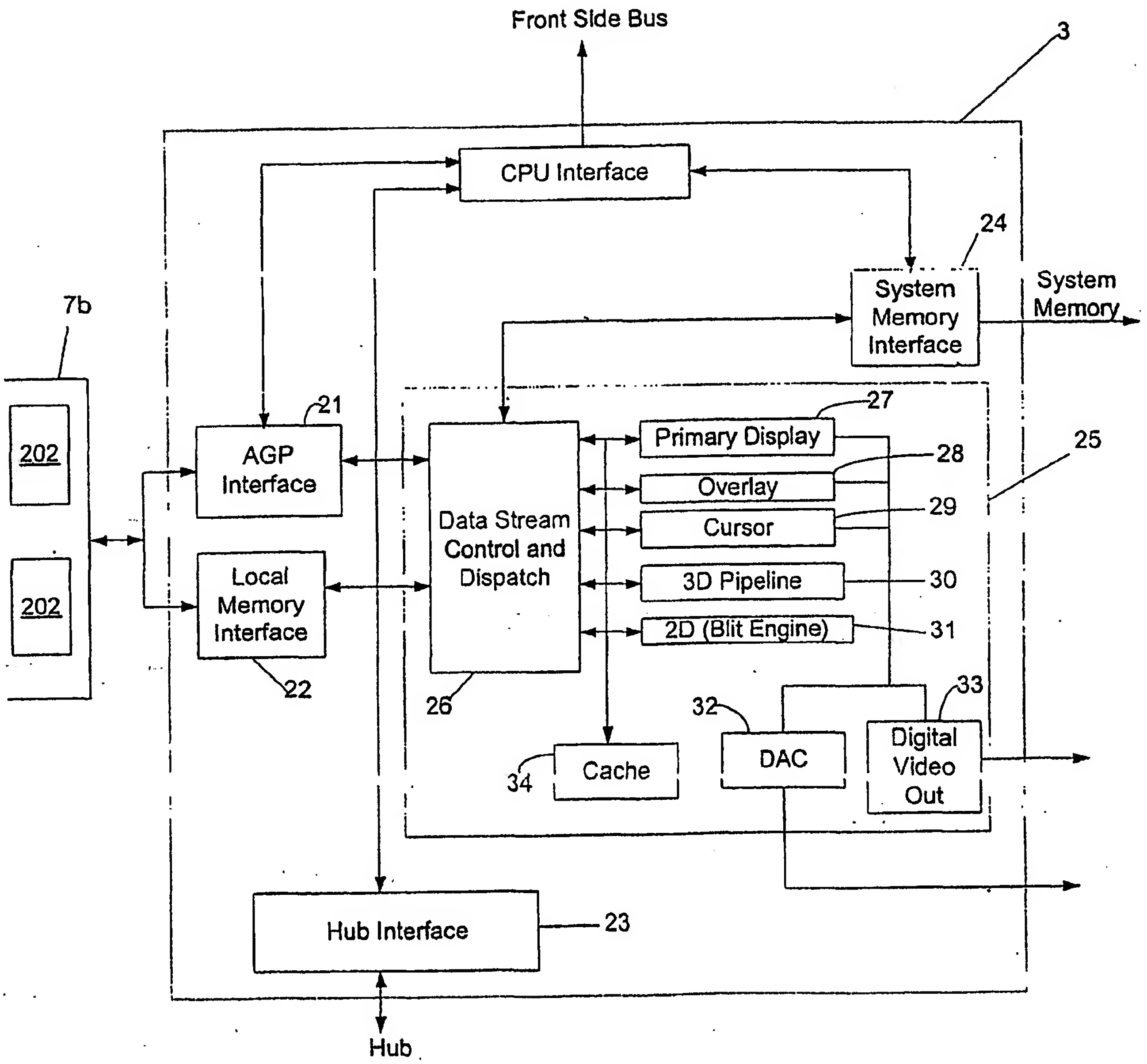

Front Side Bus
3
CPU Interface
24
System Memory Interface
System Memory
7b
202
202
21
AGP Interface
Local Memory Interface
22
27
Primary Display
28
Overlay
29
Cursor
3D Pipeline
30
2D (Blit Engine)
31
25
Data Stream Control and Dispatch
26
32
33
DAC
Digital Video Out
34
Cache
Hub Interface
23
Hub

Figure 4